Instituto Tecnológico de Estudios Superiores de Monterrey

Campus Puebla

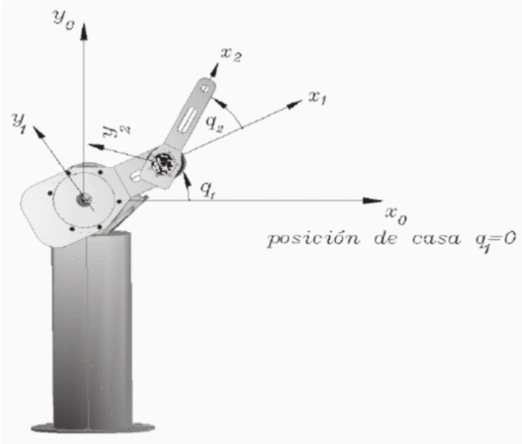
Fundamentación de Robótica (Gpo 101)

Actividad 4 (Modelado de Energía Cinética)

Alumno
José Diego Tomé Guardado A01733345
Pamela Hernández Montero A01736368
Victor Manuel Vázquez Morales A01736352
Fernando Estrada Silva A01736094

Fecha de entrega
Miércoles 6 de Marzo de 2024

Ejercicio 1 - Robot planar de 2gdl :



Primeramente vamos a modelar nuestro diseño del robot planar de 2gdl, como podemos observar y ya sabemos tenemos dos juntas rotacionales. Por lo que tendremos solo implementadas las matrices de transformación para el eje Z que donde tiene el movimiento rotacional, además de declarar siempre la posición tomando en cuenta el coseno y el seno que podemos ver que se forma.

De esta manera vamos a implementar la explicación del siguiente código para encontrar la energía cinética de cada eslabón y también la energía cinética general de las dos juntas. Vamos a explicar los puntos más importantes del código para poder hacer nuestro análisis y también hacer hincapié en lo que se modificó.

1. **Velocidades lineales y angulares para cada articulación**

Para las velocidades vamos a tomar en cuenta el número de eslabones que vamos a tener por lo que en este primer ejercicio tenemos dos eslabones por lo que el eslabón 1 se calculará de la siguiente manera:

```
%Eslabón 1
%Calculamos el jacobiano lineal y angular de forma analítica
Jv_a1(:,GDL-1)=PO(:,:,GDL-1);
Jw_a1(:,GDL-1)=PO(:,:,GDL-1);
for k= 1:GDL-1
   if RP(k)==0
      %Para las juntas de revolución
       try
           Jv_a1(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL-1)-PO(:,:,k-1));
           Jw_a1(:,k)= RO(:,3,k-1);
       catch
           Jv_a1(:,k)= cross([0,0,1], PO(:,:,GDL-1));%Matriz de rotación de 0
con respecto a 0 es la Matriz Identidad
           Jw_a1(:,k)=[0,0,1];%Si no hay matriz de rotación previa se obtiene
la Matriz identidad
        end
```

Por lo que podemos observar en este caso tendremos el GDL-1 debido a qué vamos a tener qué calcular el jacobiano lineal como el angular para el primer eslabón relacionando las velocidades lineales y angulares, en este caso tiene un menos uno debido a qué el bucle for anterior se ejecuta para k desde 1 hasta GDL-1 de esta manera se está iterando a través de los grados de libertad antes de el ultimo y tambien se refiere a que es el penúltimo grado de libertad.

A continuación también vamos a tener el cálculo de las velocidad lineal como angular obtenidas mediante el jacobiano, solo basta con multiplicar, para la velocidad lineal se obtiene del producto del jacobiano lineal `Jv_a` y las velocidades articulares `Qp`. Para la velocidad angular se obtiene del producto del jacobiano angular `Jw_a` y las velocidades articulares `Qp`.

```
%Obtenemos SubMatrices de Jacobianos
Jv_a1= simplify (Jv_a1);
Jw_a1= simplify (Jw_a1);
%disp('Jacobiano lineal obtenido de forma analítica');
%pretty (Jv_a);
%disp('Jacobiano ángular obtenido de forma analítica');
%pretty (Jw_a);
%Matriz de Jacobiano Completa
%disp('Matriz de Jacobiano');
Jac1= [Jv_a1;
     Jw_a1];
Jacobiano1= simplify(Jac1);
% pretty(Jacobiano);
%Obtenemos vectores de Velocidades Lineales y Angulares
%disp('Velocidad lineal obtenida mediante el Jacobiano lineal del Eslabón
1');
Qp=Qp(t);
V1=simplify (Jv_a1*Qp(1));
%pretty(V1);
% disp('Velocidad angular obtenida mediante el Jacobiano angular del Eslabón
1');
W1=simplify (Jw_a1*Qp(1));
% pretty(W1);
```

De la misma forma ya vamos a tener calculado lo del primer eslabón debido a que se hizo anteriormente de este pero como los jacobianos generales ya de todo el programa porque se considera ya como última articulación:

```
for k= 1:GDL
   if RP(k)==0
      %Para las juntas de revolución
       try
           Jv_a(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL)-PO(:,:,k-1));
           Jw_a(:,k)= RO(:,3,k-1);
       catch
             Jv_a(:,k)= cross([0,0,1], PO(:,:,GDL));%Matriz de rotación de 0
con respecto a 0 es la Matriz Identidad, la posición previa también será 0
            Jw_a(:,k)=[0,0,1];%Si no hay matriz de rotación previa se obtiene
la Matriz identidad
        end
```

Por lo que vemos este GDL no le restamos ningún número ni nada, lo dejamos igual al que teníamos en los anteriores programas de las actividades, en este caso se referiría a GDL-0 que tendría como explicación que es la última articulación o eslabón quedando de esa manera expresado para poder ocupar los valores de sus velocidades para el cálculo de la energía cinética. Y de la misma manera tenemos el cálculo de la velocidad lineal como angular respecto a lo qué indicará este segundo eslabón:

```
%Obtenemos SubMatrices de Jacobianos
Jv_a= simplify (Jv_a);
Jw_a= simplify (Jw_a);
%disp('Jacobiano lineal obtenido de forma analítica');
%pretty (Jv_a);
%disp('Jacobiano ángular obtenido de forma analítica');
%pretty (Jw_a);
%Matriz de Jacobiano Completa
%disp('Matriz de Jacobiano');
Jac= [Jv_a;
     Jw_a];
Jacobiano= simplify(Jac);
%pretty(Jacobiano);
%Obtenemos vectores de Velocidades Lineales y Angulares
% disp('Velocidad lineal obtenida mediante el Jacobiano lineal');
V=simplify (Jv_a*Qp);
% pretty(V);
% disp('Velocidad angular obtenida mediante el Jacobiano angular');
W=simplify (Jw_a*Qp);
%     pretty(W);
```

2. Análisis por articulación

Tendremos ahora que encontrar la matriz homogénea de transformación, haciendo un análisis podemos darnos cuenta de que solo contamos con dos articulaciones, por lo que obtenerla resulta algo sencillo. Pero una mejor idea es multiplicar todas las matrices homogéneas locales, las cuales toman como origen el extremo final de la anterior articulación y de esta forma podemos decir que tendremos dos matrices homogéneas locales.

Teniendo por consiguiente nuestros matrices de posición P dónde lo haremos de manera local por cada una de nuestras articulaciones. También declaramos nuestras matrices de rotación `R` para cada articulación y por último tendremos un vector de ceros.

```
%Articulación 1
%Posición de la articulación 1 respecto a 0
P(:,:,1)= [l1*cos(th1); l1*sin(th1);0];
%Matriz de rotación de la junta 1 respecto a 0....
R(:,:,1)= [cos(th1) -sin(th1)  0;
          sin(th1)  cos(th1)  0;
          0         0         1];
%Articulación 2
%Posición de la articulación 1 respecto a 0
P(:,:,2)= [l2*cos(th2); l2*sin(th2);0];
%Matriz de rotación de la junta 1 respecto a 0....
```

```
R(:,:,2)= [cos(th2) -sin(th2)  0;
           sin(th2)  cos(th2)  0;
           0         0         1];

Vector_Zeros = zeros(1,3);
```

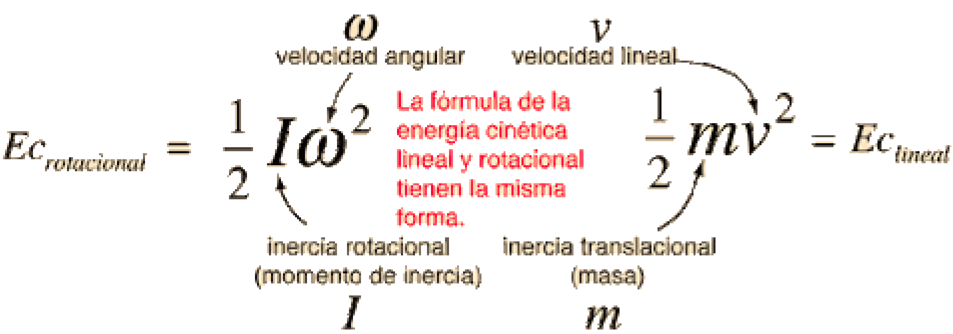
3. Matrices de inercia para cada articulación

A partir de este punto se obtiene la distancia del origen del eslabón hasta su centro de masa. Es decir, se calcula la posición del centro de masa del eslabón 1 con respecto al sistema de referencia inercial. Por eso mismo, se incluye la matriz de inercia, las cuales se encargan de describir como la masa está distribuida en torno a sus ejes de referencia. Las *Ixx* representan los momentos principales de inercia, siendo la resistencia a la rotación alrededor de los ejes principales.

```
P01=subs(P(:,:,1)/2, l1, lc1); %La función subs sustituye l1 por lc1 en
                               %la expresión P(:,:,1)/2
P12=subs(P(:,:,2)/2, l2, lc2);
%Creamos matrices de inercia para cada eslabón
I1=[Ixx1 0 0;
   0 Iyy1 0;
   0 0 Izz1];
I2 = [Ixx2 0 0;
     0 Iyy2 0;
     0 0 Izz2];
```

4. Calculamos energía cinética para cada eslabón

Ahora, pasando a la sección del cálculo de la energía cinética, este proceso debe realizarse de forma independiente para cada eslabón. V1:Total representa la velocidad lineal total en el primer eslabón, considerando la velocidad generalizada (V1) y la contribución cruzada por la rotación. Posteriormente se utiliza la ecuación de energía cinética para obtenerla, además de agregarle la velocidad lineal y angular previamente calculadas. En este caso, debido a que el robot tiene tanto movimiento lineal como rotacional, se considera que la energía cinética total es la suma de la energía cinética lineal, asociada al movimiento lineal del robot a través del espacio, y la energía cinética rotacional, asociada al movimiento de rotación del robot alrededor de su centro de masa.



Si analizamos la fórmula de la energía cinética total, podemos observar que refleja la contribución de ambos tipos de movimiento al total de energía del sistema.

```
%Eslabon1
V1_Total= V1+cross(W1,P01);
K1= (1/2*m1*(V1_Total))'*((V1_Total)) + (1/2*W1)'*(I1*W1);
disp('Energía Cinética en el Eslabón 1');
K1= simplify (K1);
pretty (K1)


%Eslabon 2
V2_Total= V+cross(W,P12);
K2= (1/2*m2*(V2_Total))'*((V2_Total)) + (1/2*W)'*(I2*W);
disp('Energía Cinética en el Eslabón 2');
K2= simplify (K2);
pretty (K2)
K_Total= simplify (K1+K2)
pretty (K_Total)
%K_Total= simplify (K1+K2);
```

Resultados

Energía cinética efectuada en eslabón 1

```
Energía Cinética en el Eslabón 1
              2            2     ______           __          2             2
Izz1 |th1p(t)|    |th1p(t)|  cos(th1(t) - th1(t)) m1 (l1 |lc1|  + 2 lc1 |l1| ) (2 l1 + lc1)
-------------- + -------------------------------------------------------------------------
       2                                           8 l1 lc1

Energía Cinética en el Eslabón 2
/
| __ /                                                              lc2 sin(th2(t)) #1 \
| m2 | th1p(t) (l1 sin(th1(t)) + l2 sin(#4)) + l2 sin(#4) th2p(t) + ------------------ |
\    \                                                                       2         /

   /                                                                    ______  ___   \ \             / _______   _______ \
   | _______          __    ______  __    _______         __  sin(th2(t)) lc2 #2 | |             | th1p(t)   th2p(t) |
   | th1p(t) (sin(#3) l2 + sin(th1(t)) l1) + th2p(t) sin(#3) l2 + ------------------ | |/2 + Izz2 #1 | ------- + ------- |
   \                                                                     2         / /             \    2         2    /

     /
```

Energía cinética efectuada en eslabón 2

```
Energía Cinética en el Eslabón 2
/
| __ /                                                              lc2 sin(th2(t)) #1 \
| m2 | th1p(t) (l1 sin(th1(t)) + l2 sin(#4)) + l2 sin(#4) th2p(t) + ------------------ |
\    \                                                                       2         /

   /                                                                    ______  ___   \ \             / _______   _______ \
   | _______          __    ______  __    _______         __  sin(th2(t)) lc2 #2 | |             | th1p(t)   th2p(t) |
   | th1p(t) (sin(#3) l2 + sin(th1(t)) l1) + th2p(t) sin(#3) l2 + ------------------ | |/2 + Izz2 #1 | ------- + ------- |
   \                                                                     2         / /             \    2         2    /

     /
     | __ /                                                              lc2 cos(th2(t)) #1 \
   + | m2 | th1p(t) (l1 cos(th1(t)) + l2 cos(#4)) + l2 cos(#4) th2p(t) + ------------------ |
     \    \                                                                       2         /

   /                                                                                 \ \
```

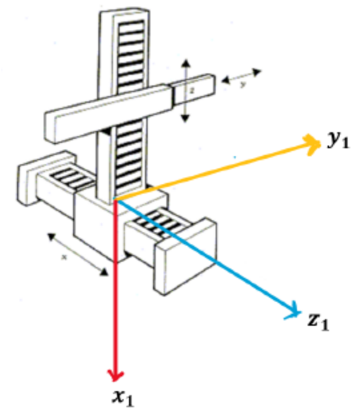
```
where

  #1 == th1p(t) + th2p(t)

        _______   _______
  #2 == th1p(t) + th2p(t)

        ______   ______
  #3 == th1(t) + th2(t)

  #4 == th1(t) + th2(t)
```

Energía cinética en el sistema

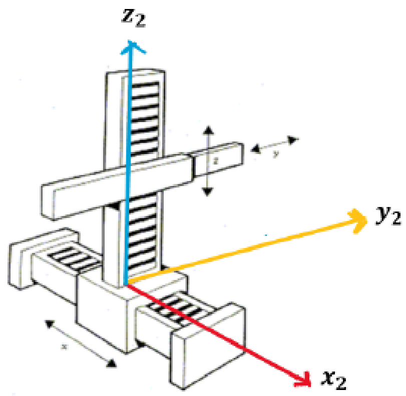
```
K_Total =

(Izz1*abs(th1p(t))^2)/2 + (conj(m2)*(th1p(t)*(l1*sin(th1(t)) + l2*sin(th1(t) + th2(t))) + l2*sin(th1(t) + th2(t))*th2p(t) + (

           /
Izz1 #5    | __ /                                                          lc2 sin(th2(t)) #1 \
------- +  | m2 | th1p(t) (l1 sin(th1(t)) + l2 sin(#4)) + l2 sin(#4) th2p(t) + ------------------ |
   2       \    \                                                                  2            /

  /                                                                   ______   ___      \ \              / _______   _______ \
  | _______          __          ______   __    _______           __  sin(th2(t)) lc2 #2 | |              | th1p(t)   th2p(t) |
  | th1p(t) (sin(#3) l2 + sin(th1(t)) l1) + th2p(t) sin(#3) l2 + ------------------ | |/2 + Izz2 #1 | ------- + ------- |
  \                                                                      2           / /              \    2         2    /

     /
     | __ /                                                         lc2 cos(th2(t)) #1 \
   + | m2 | th1p(t) (l1 cos(th1(t)) + l2 cos(#4)) + l2 cos(#4) th2p(t) + ------------------ |
     \    \                                                                 2            /


where

  #1 == th1p(t) + th2p(t)

        _______   _______
  #2 == th1p(t) + th2p(t)

        ______   ______
  #3 == th1(t) + th2(t)

  #4 == th1(t) + th2(t)

                    2
  #5 == |th1p(t)|
```

Ejercicio 2 - Robot lineal de 3GDL:

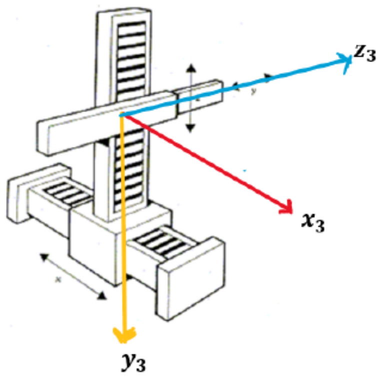
Para modelar la energía cinética para este robot, antes es necesario declarar su configuración para así obtener las respectivas matrices de transformación homogéneas. Tal y como lo habíamos implementado en la tarea 2, este robot se compone de 3 juntas prismáticas y sus matrices de rotación y traslación se declaran de la siguiente forma:




```
% Articulación 1
% Posición
P(:,:,1) = [0; 0; l1];
% Matriz de rotación
R(:,:,1) = [0 0 -1;
            0 1 0;
            1 0 0];
```




```
% Articulación 2
%Posición
P(:,:,2) = [0; 0; l2];
%Matriz de rotación
R(:,:,2) = [1 0 0;
            0 0 1;
            0 -1 0];
```




```
% Articulación 3
%Posición
P(:,:,3) = [0; 0; l3];
%Matriz de rotación
R(:,:,3) = [1 0 0;
            0 1 0;
            0 0 1];
```

Posterior a esto, el código de MATLAB se encarga de construir las matrices de transformación realizando la concatenación de las matrices de rotación, posición, vector de zeros, etc. Ahora bien, para obtener el modelado de energía cinética utilizaremos la ecuación previamente mencionada en el primer ejercicio, implementandolo de la siguiente manera:

1. Obtenemos la velocidad lineal y angular para cada articulación: Dentro de la ecuación para la energía cinética podemos observar que esta considera a la velocidad lineal y la velocidad angular. Para calcular estos valores/variables en cada una de las articulaciones utilizaremos el código del jacobiano y velocidades implementado en actividades pasadas pero considerando únicamente hasta la articulación actual, modificando el for y la indexación para que solo considere hasta el eslabón de interés. Por ejemplo, para obtener la velocidad lineal y angular para el eslabón 1 se realizan modificaciones como las siguientes:

```
for k= 1:GDL-2
    ...
    try
        J v_a1(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL-2)-PO(:,:,k-1));
    catch
        Jv_a1(:,k)= cross([0,0,1], PO(:,:,GDL-2));
end
```

```
Jv_a1= simplify (Jv_a1);
Jw_a1= simplify (Jw_a1);
```

De esta forma, podemos obtener la velocidad lineal y angular:

```
Qp=Qp(t);
V1=simplify (Jv_a1*Qp(1));
W1=simplify (Jw_a1*Qp(1));
```

2.Declaración de los vectores de posición respecto al centro de masa: Otra parte fundamental para la obtención de la energía cinética es tomar en consideración la distancia al centro de masa para cada eslabón. Dicho esto, declararemos vectores de posición respecto al centro de masa, similares a los declarados previamente pero haciendo un cambio de variables:

```
P01=subs(P(:,:,1), l1, lc1);%La función subs sustituye l1 por lc1 en
P12=subs(P(:,:,2), l2, lc2); %la expresión P(:,:,1)/2
P23=subs(P(:,:,3), l3, lc3);
```

3. Creación de las matrices de inercia para cada eslabón: Notemos que, de igual forma, la ecuación para energía cinética considera igualmente la inercia rotacional, por lo que, agregando variables simbólicas a nuestro código, procedemos a declarar las matrices de inercia utilizando matrices, esto debido a que todo nuestro código funciona de manera matricial:

```
I1=[Ixx1 0 0;
   0 Iyy1 0;
   0 0 Izz1];
I2=[Ixx2 0 0;
   0 Iyy2 0;
   0 0 Izz2];
I3=[Ixx3 0 0;
   0 Iyy3 0;
   0 0 Izz3];
```

4. Implementación de la ecuación para energía cinética: Finalmente, es momento de declarar la fórmula de energía cinética y observar los resultados que nos arroja. Sin embargo, antes de continuar con la implementación en código, es importante mencionar que se hará una modificación a la velocidad lineal. Observemos por ejemplo, la modificación realizada para el primer eslabón:

```
V1_Total= V1+cross(W1,P01);
```

Esta ecuación nos indica que la velocidad lineal total es igual a la previamente calculada más el producto cruz entre la velocidad angular y la distancia al centro de masa del eslabón, ya que la rotación puede afectar de igual forma en la velocidad lineal.

Dicho esto, finalmente, podemos declarar la ecuación de energía cinética para cada eslabón:

```
%Eslabón 1
V1_Total= V1+cross(W1,P01);
K1= (1/2*m1*(V1_Total))'*((V1_Total)) + (1/2*W1)'*(I1*W1);
disp('Energía Cinética en el Eslabón 1');
K1= simplify (K1);
pretty (K1);

%Eslabón 2
V2_Total= V2+cross(W2,P12);
K2= (1/2*m2*(V2_Total))'*((V2_Total)) + (1/2*W2)'*(I2*W2);
disp('Energía Cinética en el Eslabón 2');
K2= simplify (K2);
pretty (K2);

%Eslabón 3
V3_Total= V+cross(W,P23);
K3= (1/2*m3*(V3_Total))'*((V3_Total)) + (1/2*W)'*(I3*W);
disp('Energía Cinética en el Eslabón 3');
K3= simplify (K3);
pretty (K3);
```

Notemos que para cada eslabón se está aplicando de manera apropiada la ecuación, considerando el cuadrado de las velocidades, masas e inercias. Por último, para obtener la energía cinética de todo el sistema realizamos la sumatoria de la energía cinética para cada eslabón:

```
K_Total= simplify (K1+K2+K3);
pretty(K_Total)
```

Resultados:

```
Energía Cinética en el Eslabón 1
        2 __
|l1p(t)|  m1
------------
      2

Energía Cinética en el Eslabón 2
__          2          2
m2 (|l1p(t)|  + |l2p(t)| )
--------------------------
             2

Energía Cinética en el Eslabón 3
__          2          2          2
m3 (|l1p(t)|  + |l2p(t)|  + |l3p(t)| )
--------------------------------------
                   2

__          2          2          2           2 __   __          2          2
m3 (|l1p(t)|  + |l2p(t)|  + |l3p(t)| )   |l1p(t)|  m1   m2 (|l1p(t)|  + |l2p(t)| )
-------------------------------------- + ------------ + --------------------------
                   2                           2                     2
```